Creación de Aplicaciones Python

Lección 5: Autentificación de usuarios

eip ESCUELA INTERNACIONAL POSGRADOS

ÍNDICE

Lección 5: Autentificación de usuarios

PRESENTACIÓN Y OBJETIVOS

En esta lección aprenderemos como crear un autentificación de usuarios para dar mayor seguridad a nuestra aplicación, este proceso es único y le diferencia del resto de frameworks.

Objetivos

- Como crear un superusuario de acceso a la aplicación.
- Crear las plantillas para entrada, salida y cambio de contraseña en la aplicación.
- Gestión propia de Django para los usuarios.

1. PRIMEROS PASOS CON DJANGO ADMINISTRATION

Este framework a diferencia del resto posee un administrador para los usuarios, para poder acceder a él, tenemos que añadir en my_projectWeb --> urls.py:

```
from django.contrib import admin

urlpatterns = [

  path('admin/', admin.site.urls),]
```

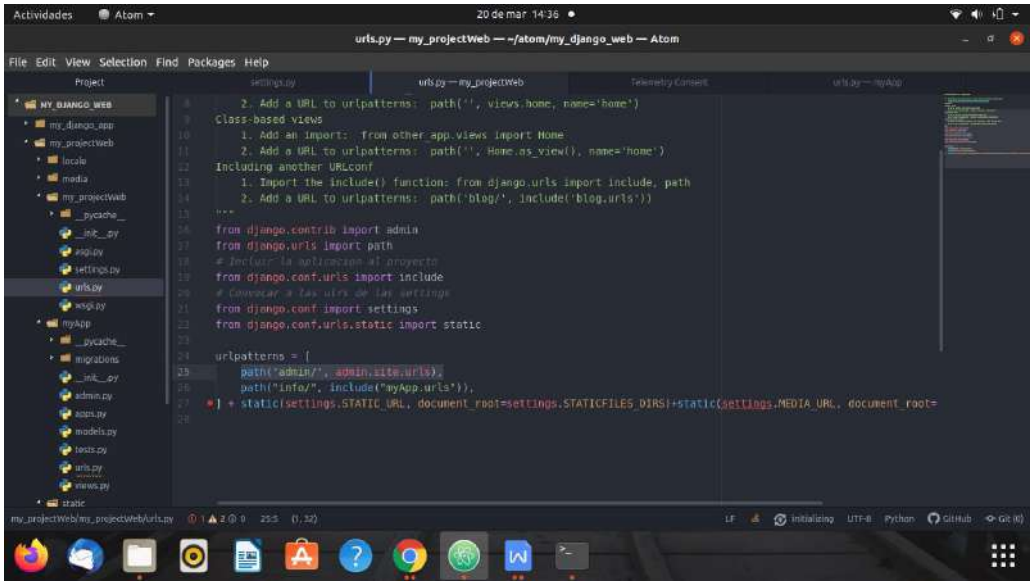


Figura 1.1: Archivo urls.py del proyecto.

Significa que accediendo a http://127.0.0.1:8000/admin/ nos mostrará la siguiente pantalla:

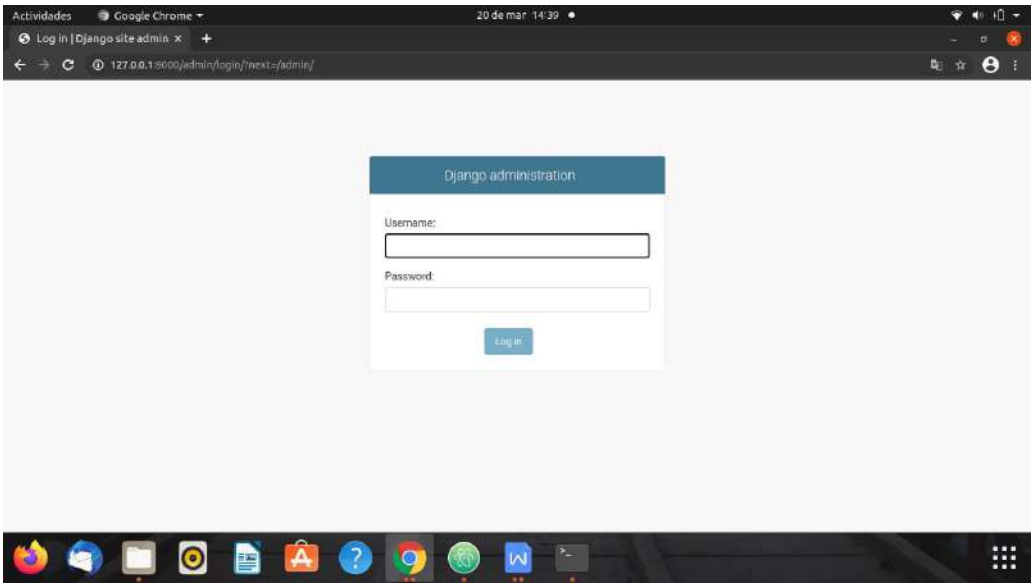


Figura 1.2: Imagen de Django Administration

Para acceder crearemos un superusuario mediante la siguiente instrucción:

```
python manage.py createsuperuser
```

User: admin

Password: cursoPython

Email: lo dejamos en blanco

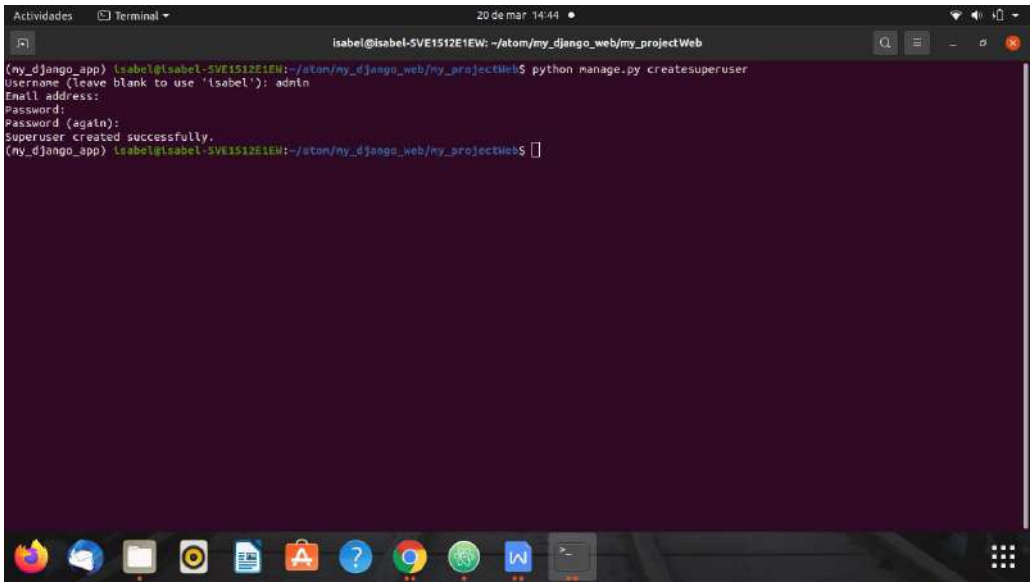


Figura 1.3: Ejemplo de crear un Superusuario

Añadimos esta instrucción a nuestro README.md:

9) Crear superusuario: python manage.py createsuperuser

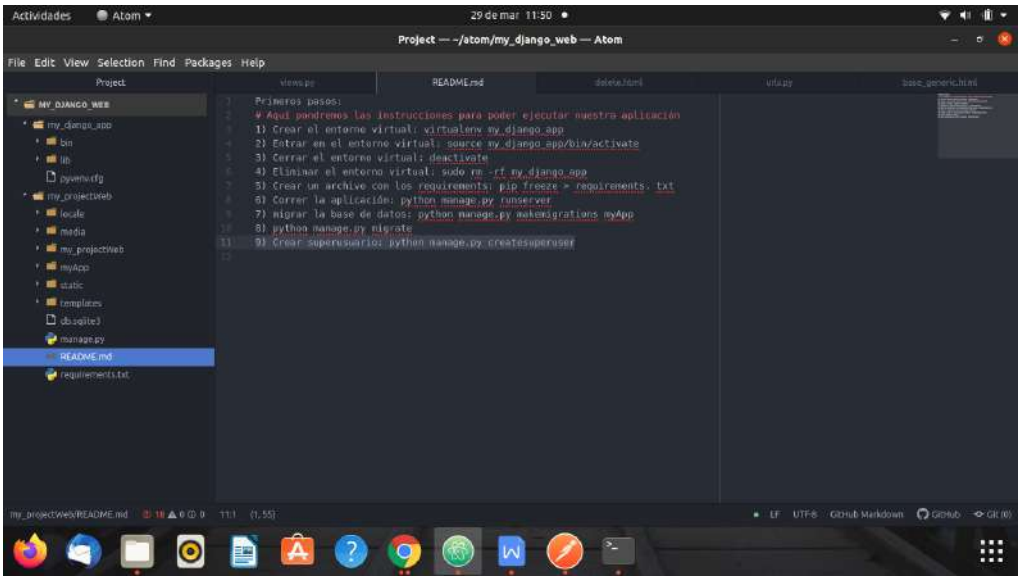


Figura 1.4: Ejemplo de README.md

Una vez creado volvemos a la página: http://127.0.0.1:8000/admin/ y ponemos las credenciales creadas.

Nos abrirá la siguiente ventana:

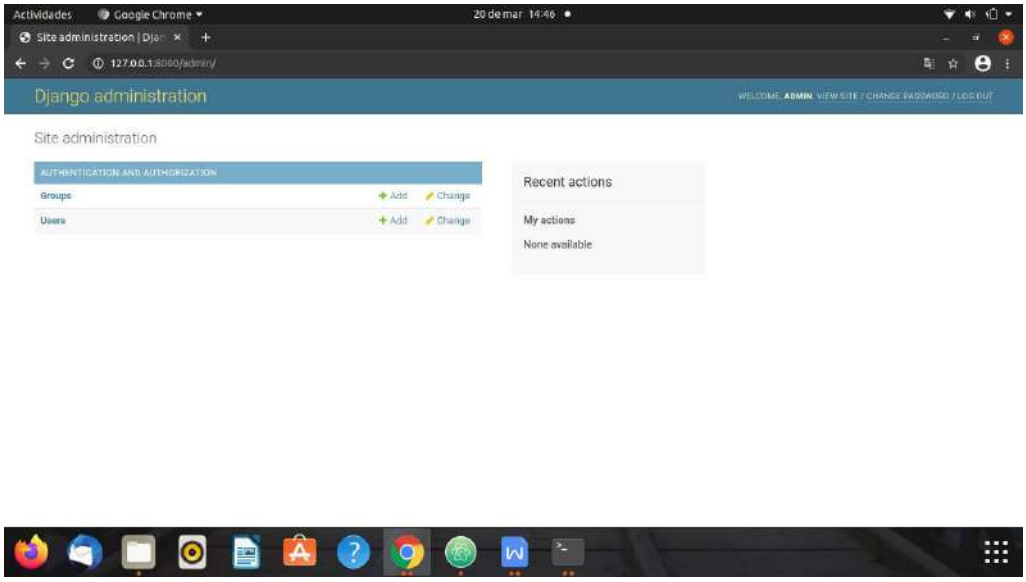


Figura 1.5: Ejemplo de Django Administration

Donde vemos dos maneras:

1. Grupos de autentificación si quieres dar unos permisos creando distintos grupos como por ejemplo un perfiles Manager, Operator o Viewer, definiendo en cada uno que puede o no hacer.
2. User es introducir los usuarios en la aplicación a través de esta ventana.

En este caso solo tenemos un usuario que es el que hemos creado nosotros como admin

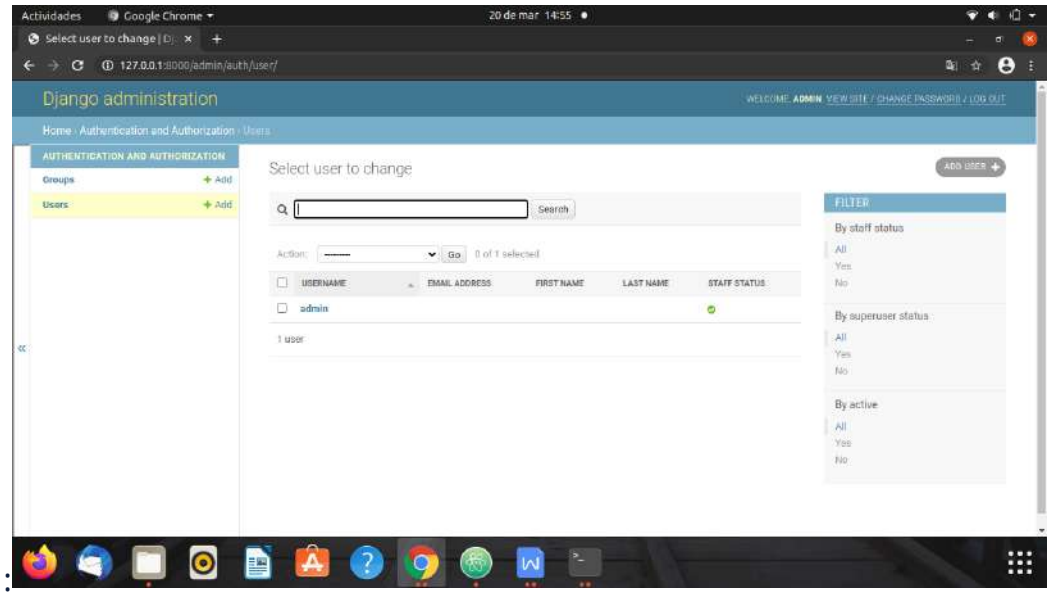


Figura 1.6: Ejemplo de Django Administration- USERS

2. CREAR UN LOGIN DE ACCESO

Ahora vamos habilitar esta opciones de logueo en nuestra aplicación:

Primero vemos si tenemos las siguientes líneas en las settings.py:

```
INSTALLED_APPS = [
    ...
    'django.contrib.auth',  # Permite la autentificación de los usuarios
    'django.contrib.contenttypes',    # Django permite los permisos de esos usuarios
    ....]
MIDDLEWARE = [
    ...
    'django.contrib.sessions.middleware.SessionMiddleware',      #Gestion de sesiones
    ...
    'django.contrib.auth.middleware.AuthenticationMiddleware',         #Asociar usuarios a las sesiones
....]
```

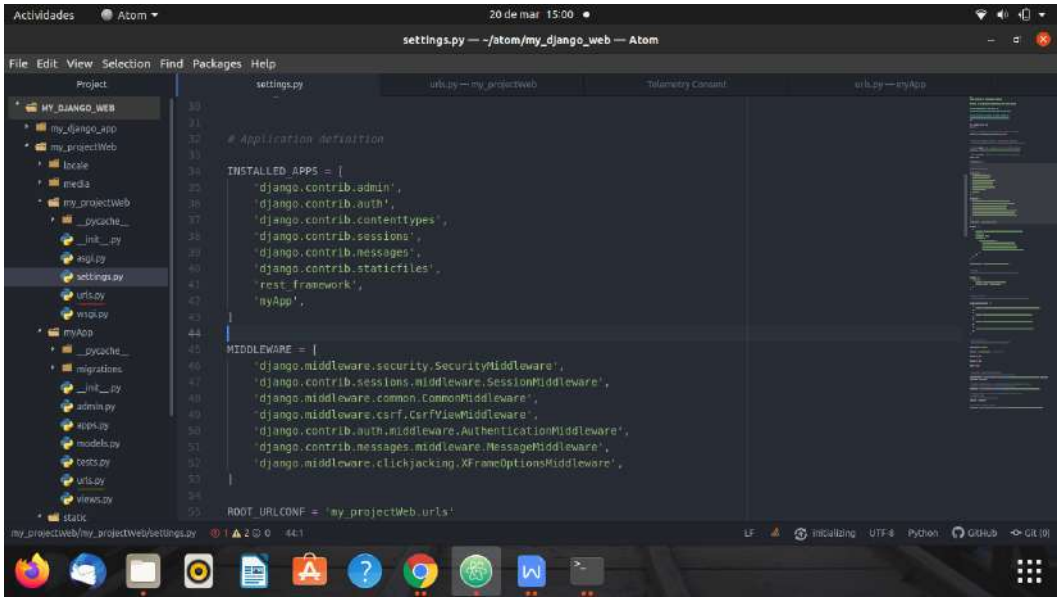


Figura 2.1: Comprobación de settings.py

Comprobamos que es así.

Vamos al archivo my_projectWeb--> my_projectWeb-->urls.py

Y ponemos la siguiente línea:

```
urlpatterns = [
...,
path('accounts/', include('django.contrib.auth.urls')), --> Para gestión de cuentas de usuarios
]
```

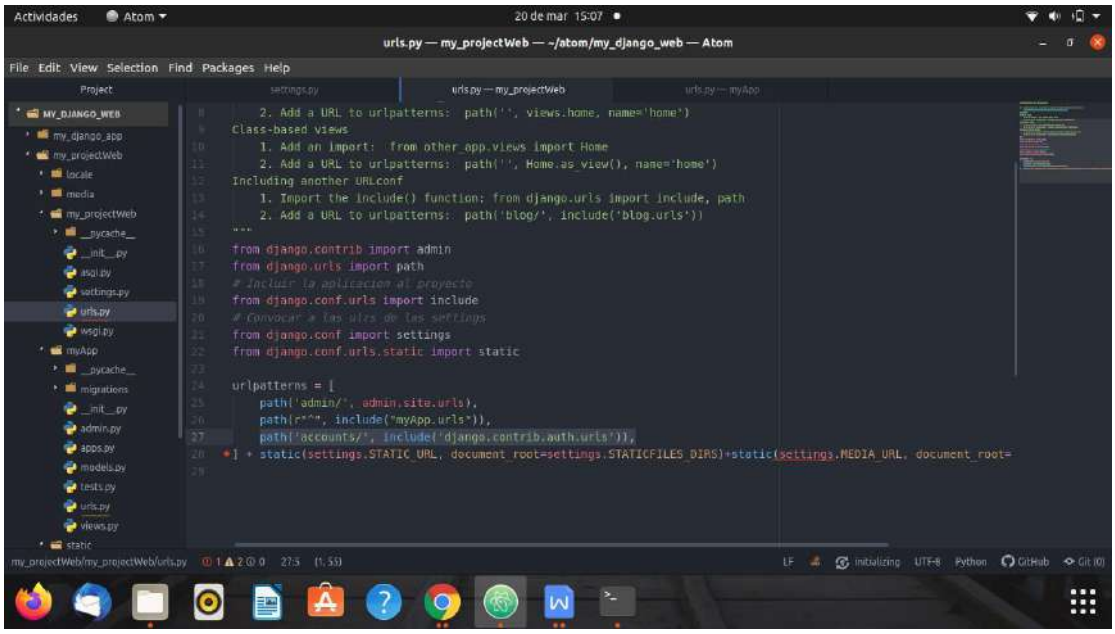


Figura 2.2: Ejemplo de urls.py del proyecto

Ahora abrimos en el navegador la siguiente página: http://127.0.0.1:8000/accounts/

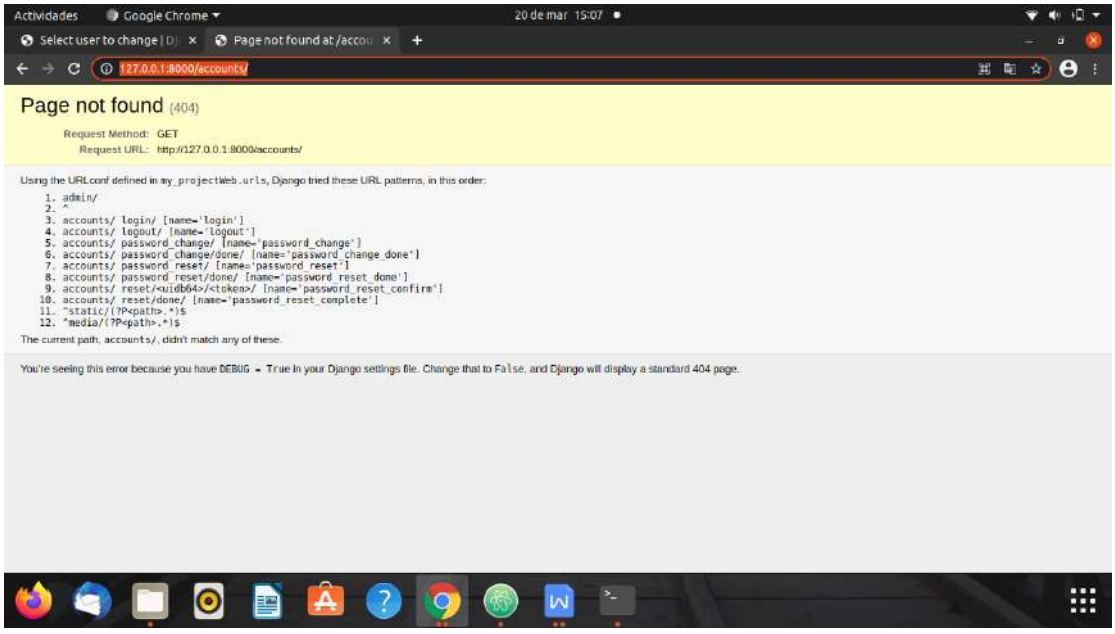


Figura 2.3: Ejemplo urls disponibles del proyecto

Vemos que Django ya incluye urls definidas para hacer el login, cambios de contraseñas, resetear el usuarios, etc.

Si ponemos una de esas Urls: http://127.0.0.1:8000/accounts/login/

Nos da un error de que todavía no tenemos creada una página de inicio:

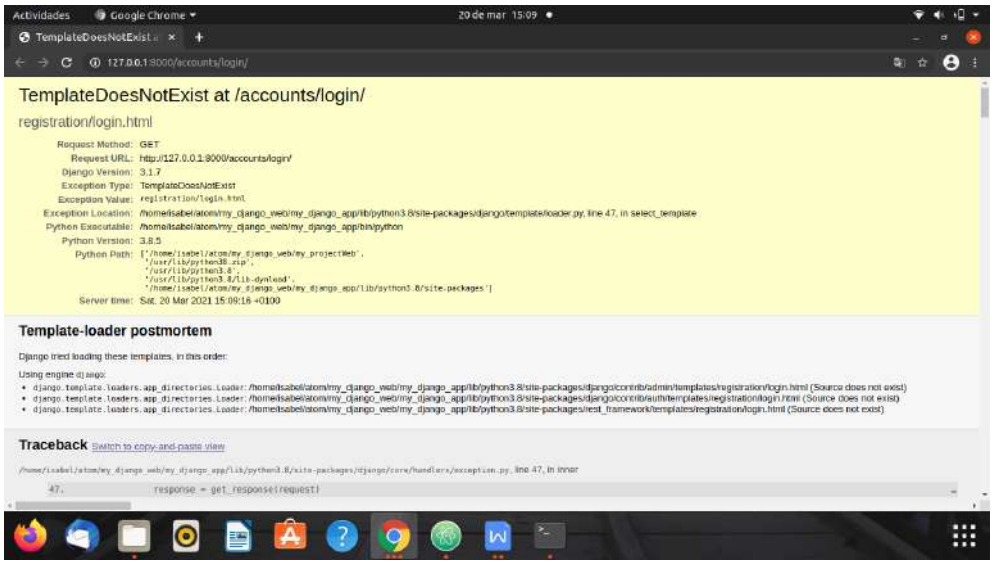


Figura 2.4: Ejemplo error por no tener plantilla HTML disponible

Añadimos también en settings.py:

```
TEMPLATES = [ {...
        'DIRS':  [
            os.path.join(BASE_DIR, 'templates'),
        ],
        'APP_DIRS': True,
        ...}, },]
```

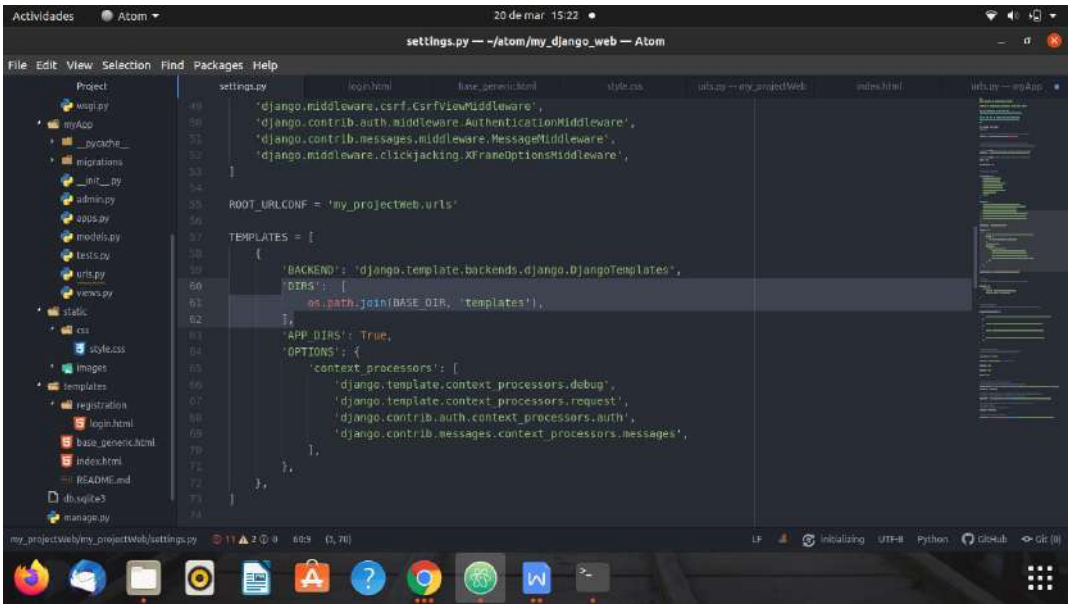


Figura 2.5: En la imagen se muestra como nombrar la ubicación de las plantillas en el proyecto

Crearemos la plantilla de la cuál van a hacerse las demás. A esa plantilla la llamaremos basic_generic.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  {% block title %}<title>Django Course</title>{% endblock %}
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link                                                    rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></scri
pt>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></sc
ript>
  <!-- Add additional CSS in static file -->
  {% load static %}
  <link rel="stylesheet" href="{% static 'css/styles.css' %}">
</head>
<body>
    <div class="container-fluid">
        <div class="row">
            <div class="col-sm-2">
            {% block sidebar %}
            <ul class="sidebar-nav" style="margin-top: 20px;">
                <li><a href="{% url '#' %}">Home</a></li>
                <li><a href="{% url '#' %}">Iris Data</a></li>
                <li><a href="{% url '#' %}">Insert Data</a></li>
                <li><a href="{% url '#' %}">Update Data</a></li>
                <li><a href="{% url '#' %}">Delete Data</a></li>
            </ul>
            {% endblock %}
        </div>
        <div class="col-sm-10 ">
        {% block content %}{% endblock %} --> la información de las plantillas
siguientes será alojada aquí
        </div>
    </div>
</body>
</html>
```

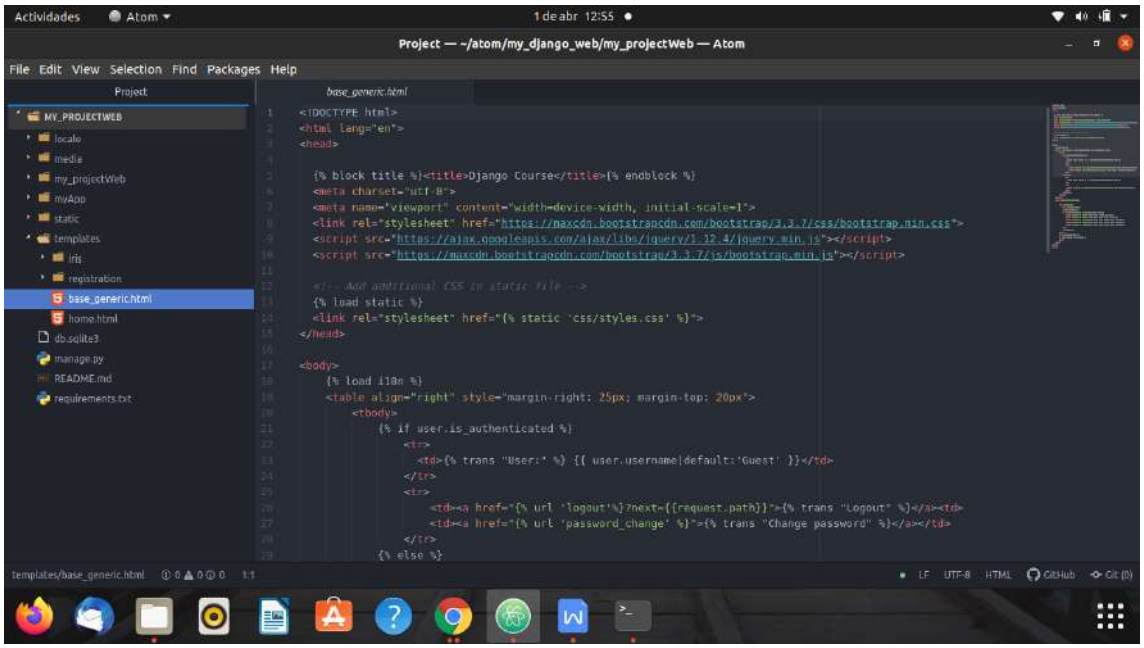


Figura 2.6: Plantilla basic_generic.html

También creamos la plantilla de home.html:

```
{% extends "base_generic.html" %} --> se insertará en la plantilla base_generic


{% block content %}

<h1>Django Home</h1>

  <p>Welcome to <em>Django application</em>, a very basic Django website developed.</p>

<h2>Dynamic content</h2>


  <p>The data has the following record counts:</p>

  <ul>

    <li><strong>Sepal length in cm:</strong> {{ sepal_length }}</li>

    <li><strong>sepal width in cm:</strong> {{ sepal_width }}</li>

    <li><strong>Petal length in cm:</strong> {{ petal_length }}</li>

    <li><strong>Petal width in cm:</strong> {{ petal_width }}</li>

    <li><strong>Class:</strong> {{ class }}</li>

  </ul>

{% endblock %}
```

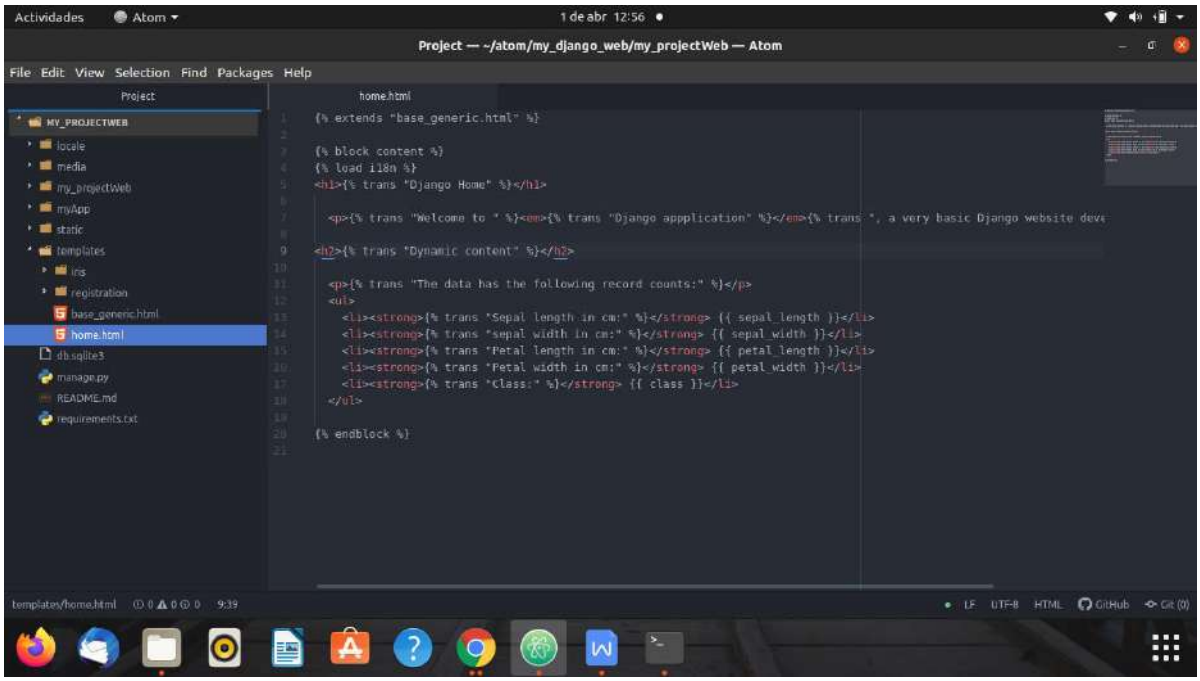


Figura 2.7: Plantilla home.html

Definimos la función en myApp --> views.py:

```
from django.shortcuts import render

def home(request):

    return
render(request,'home.html',context={'sepal_length':3,'sepal_width':2,'petal_le
ngth':2,'petal_width':2,'class':"Iris Setosa"},)
```

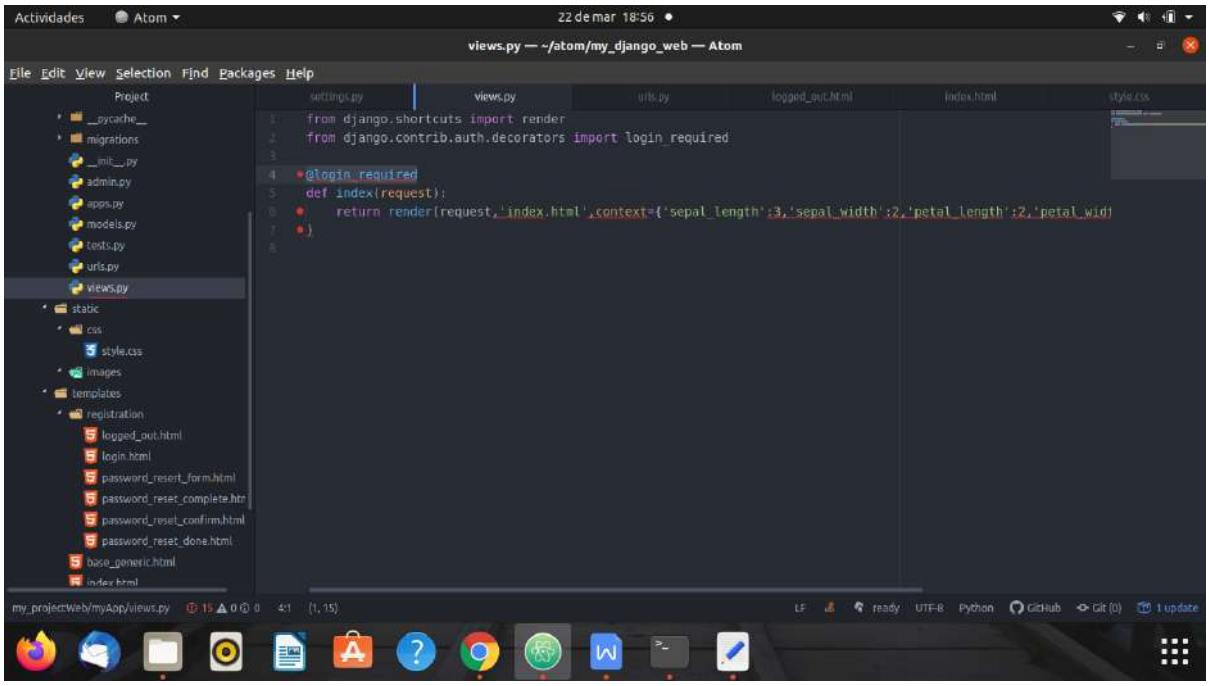


Figura 2.8: Ejemplo de urls.py en la aplicación Django

Y definimos la url en myApp --> urls.py:

```
from django.conf.urls import url, include
from myApp import views

urlpatterns = [
  url(r'^home/', views.home, name='home'),]
```

Creamos una carpeta en templates llamada registration donde pondremos el login.html que será nuestra plantilla html para el acceso de los usuarios:

My_projectWeb --> templates--> registration-->login.html

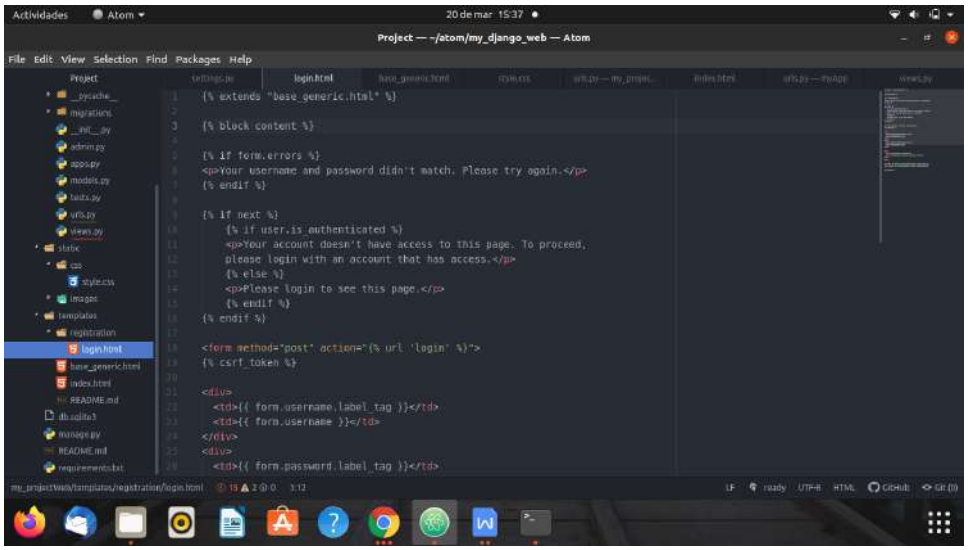


Figura 2.9: Plantilla login.html

```
{% extends "base_generic.html" %}
{% block content %}
{% if form.errors %}
<p>Your username and password didn't match. Please try again.</p>
{% endif %}
{% if next %}
    {% if user.is_authenticated %}
    <p>Your account doesn't have access to this page. To proceed,
    please login with an account that has access.</p>
    {% else %}
    <p>Please login to see this page.</p>
    {% endif %}
{% endif %}
<form method="post" action="{% url 'login' %}">
{% csrf_token %}
```

```
<div>
  <td>{{ form.username.label_tag }}</td>
  <td>{{ form.username }}</td>
</div>
<div>
  <td>{{ form.password.label_tag }}</td>
  <td>{{ form.password }}</td>
</div>

<div>
  <input type="submit" value="login" />
  <input type="hidden" name="next" value="{{ next }}" />
</div>
</form>

{# Assumes you setup the password_reset view in your URLconf #}
<p><a href="{% url 'password_reset' %}">Lost password?</a></p>

{% endblock %}
```

Volvemos a: http://127.0.0.1:8000/accounts/login/

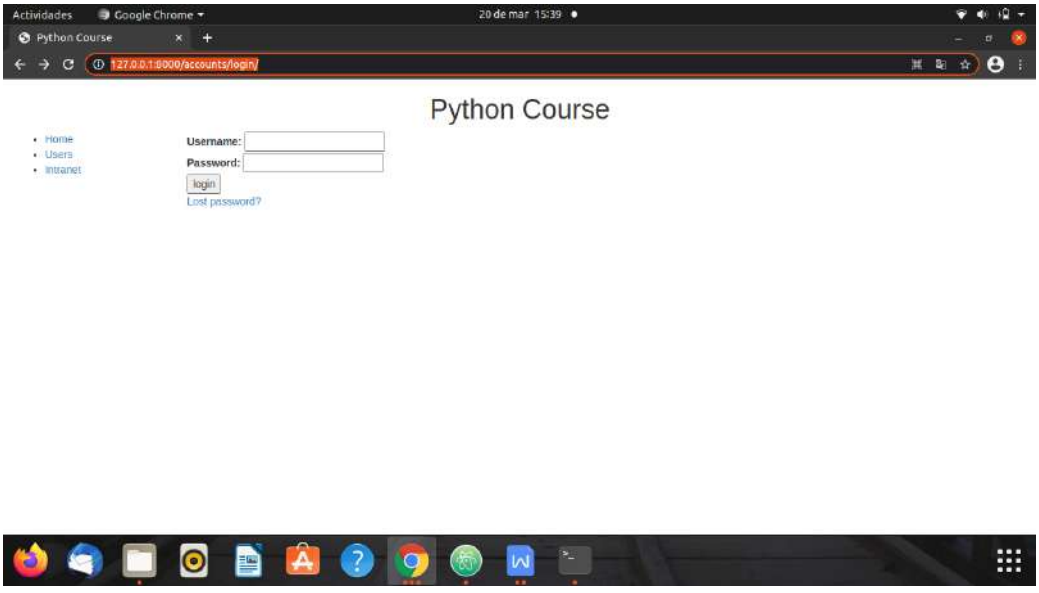


Figura 2.10: Imagen de la pantalla principal de Logueo

Vemos que nos ha creado el acceso para nuestros usuarios.

Ponemos la plantilla para Log out:

```
{% extends "base_generic.html" %}

{% block content %}

<p>Logged out!</p>

<a href="{% url 'login'%}">Click here to login again.</a>

{% endblock %}
```

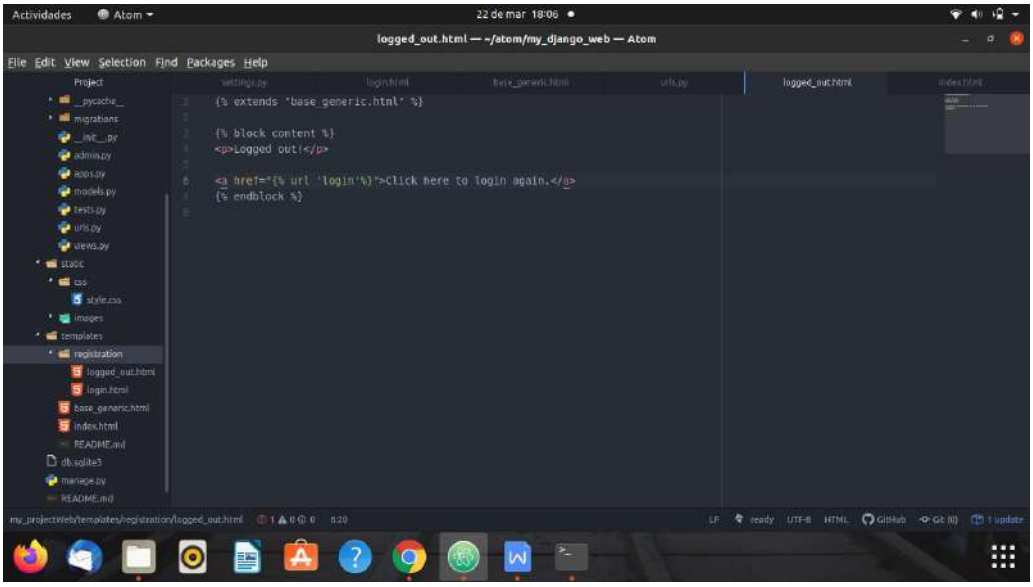


Figura 2.11: Plantilla logout.html

Probamos la url: http://127.0.0.1:8000/accounts/logout/

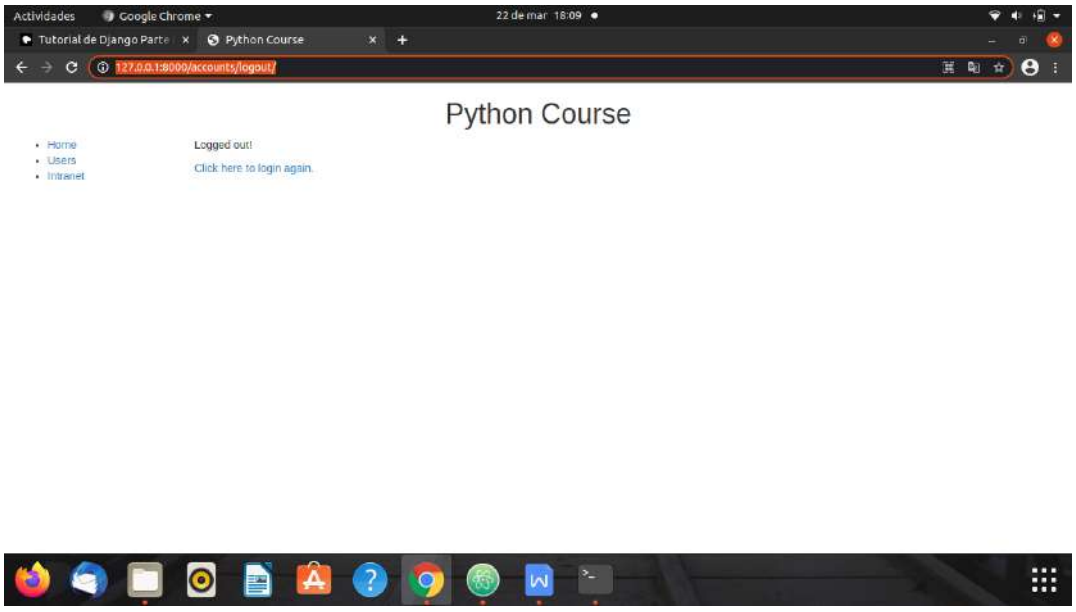


Figura 2.12: Imagen de salida de la aplicación

En el archivo base_generic.html tenemos una parte de código para ver si esta autentificado o el usuario:

```
<table align="right" style="margin-right: 25px; margin-top: 20px">
        <tbody>
            {% if user.is_authenticated %}
                <tr>
                  <td>User: {{ user.username|default:'Guest' }}</td>
                </tr>
                <tr>
                    <td><a href="{% url
'logout'%}?next={{request.path}}">Logout</a><td>
                </tr>
            {% else %}
                <tr>
                  <td>User:  {{ user.username|default:'Guest' }}</td>
                </tr>
                <tr>
                  <td><a href="{% url
'login'%}?next={{request.path}}">Login</a></td>
                </tr>
            {% endif %}
        </tbody>
</table>
```

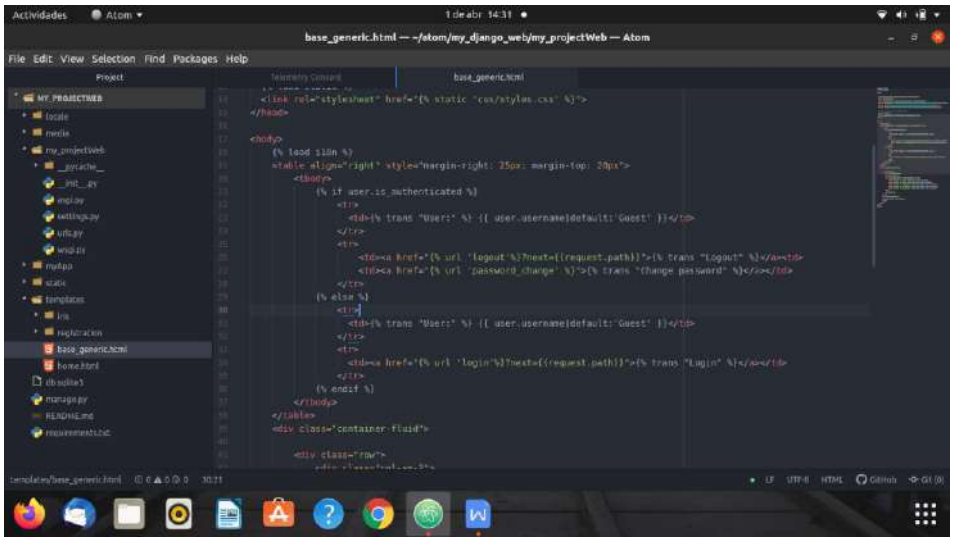


Figura 2.13: Archivo basic_generic donde ponemos la autentificación del usuario

También podemos limitar el acceso a la página mediante el uso de la librería:

```
from django.contrib.auth.decorators import login_required
```

Pondremos encima de la función:

```
@login_required
```

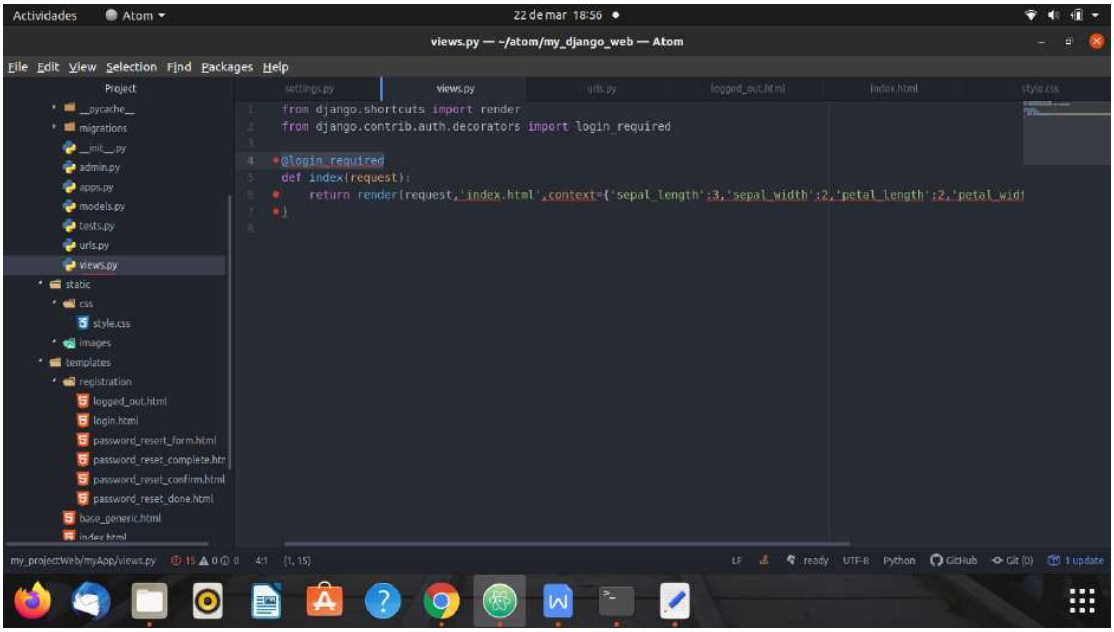


Figura 2.14: Archivo views.py de la aplicación donde se muestra el decorador para impedir la entrada sin autentificar.

Dónde si el usuario no está autentificado no podrá acceder a la página.

3. Cambio de Password

Para cambiar la contraseña ponemos la plantilla en templates --> registration --> password_change_form.html:

```
{% extends "base_generic.html" %}
{% block content %}
<h2>Change password</h2>

<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Change">
</form>
<a href="{% url 'home' %}">Back to dashboard</a>
{% endblock %}
```

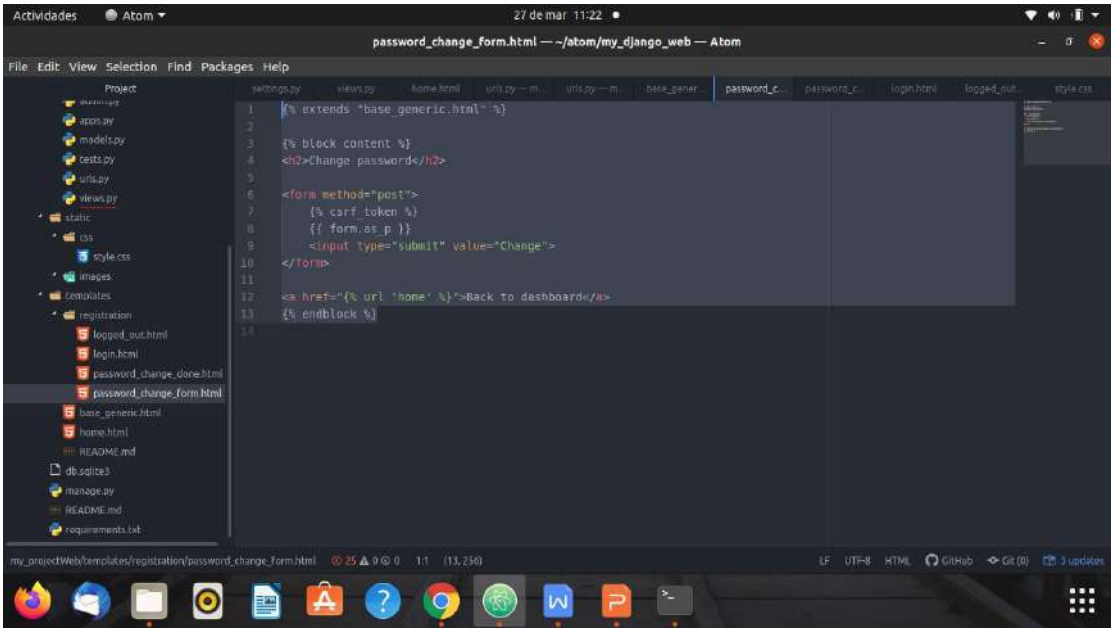


Figura 3.1: Plantilla password_change_form.htm

Una vez cambiada la contraseña creamos la plantilla de que se ha hecho con éxito para ello en templates --> registration --> password_change_done.html:

```
{% extends "base_generic.html" %}
{% block content %}
<h2>Password changed</h2>
<a href="{% url 'home' %}">Back to dashboard</a>
{% endblock %}
```

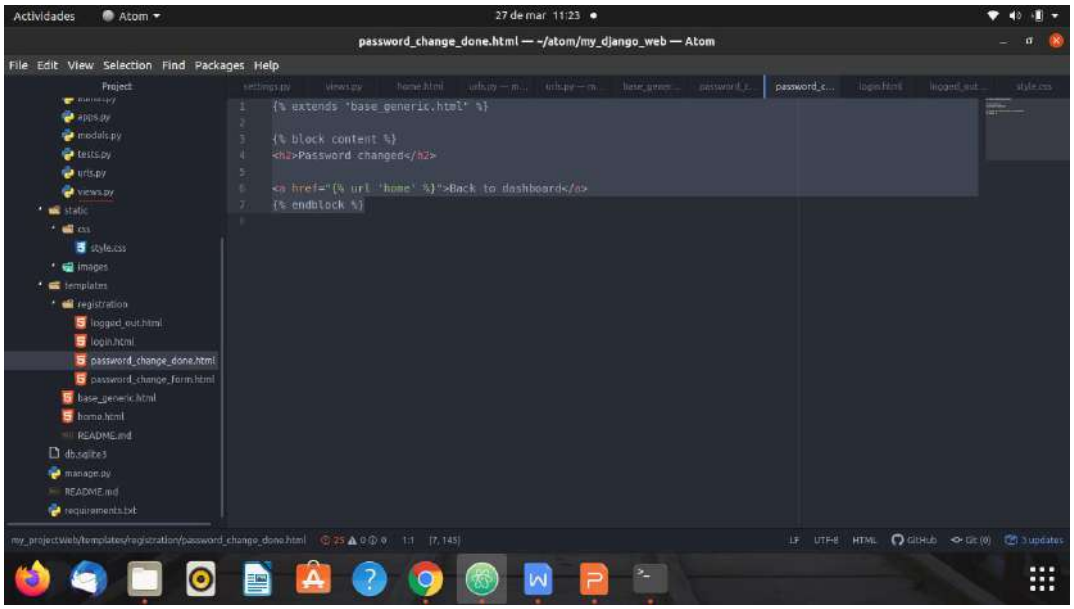


Figura 3.2: Plantilla password_change_done.htm

Añadimos en el archivo Base (Base_generic.html) la posibilidad de cambiar la contraseña:

```
<a href="{% url 'password_change' %}">Change password</a>
```

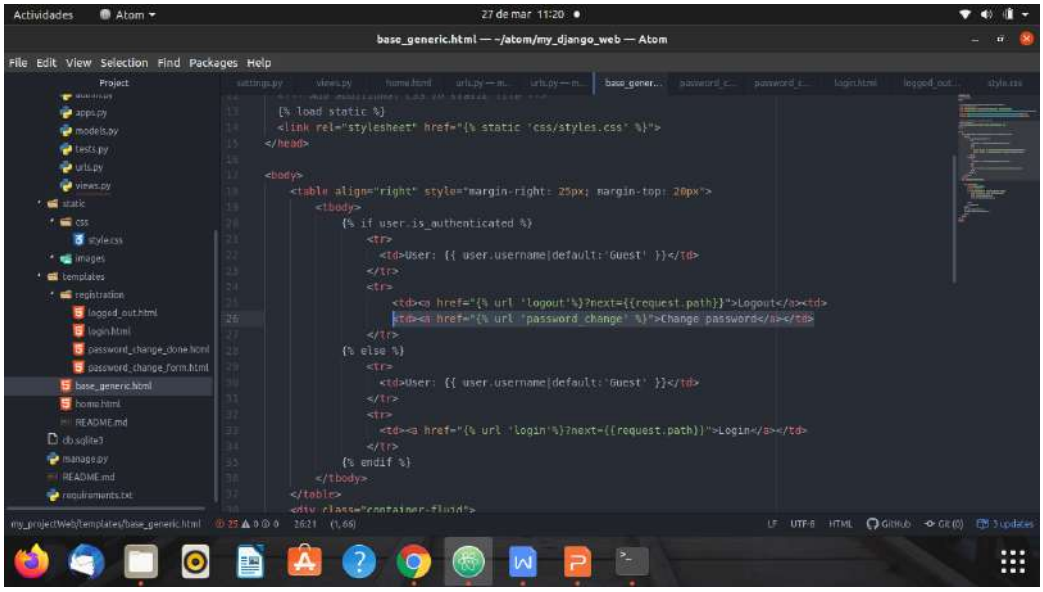


Figura 3.3: Plantilla password_change_done.htm

Al recargar la página veremos que tenemos un botón nuevo al lado de logout Changed password:

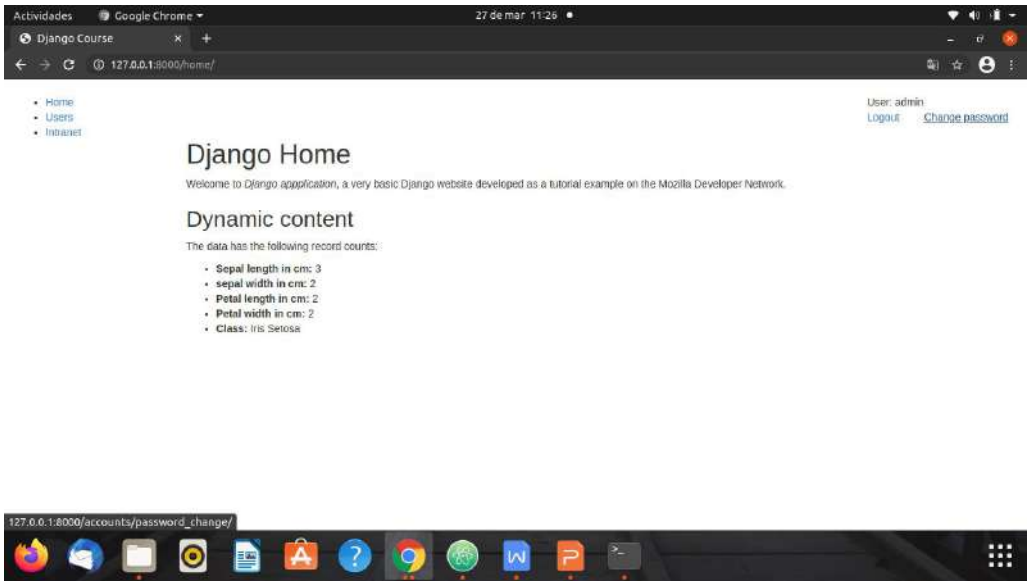


Figura 3.4: Imagen en la que se muestra la página Inicio donde en la esquina derecha arriba se ve la autentificación y los botones de salida y cambio de contraseña.

Si pulsamos nos abrirá la siguiente página:

http://127.0.0.1:8000/accounts/password_change/

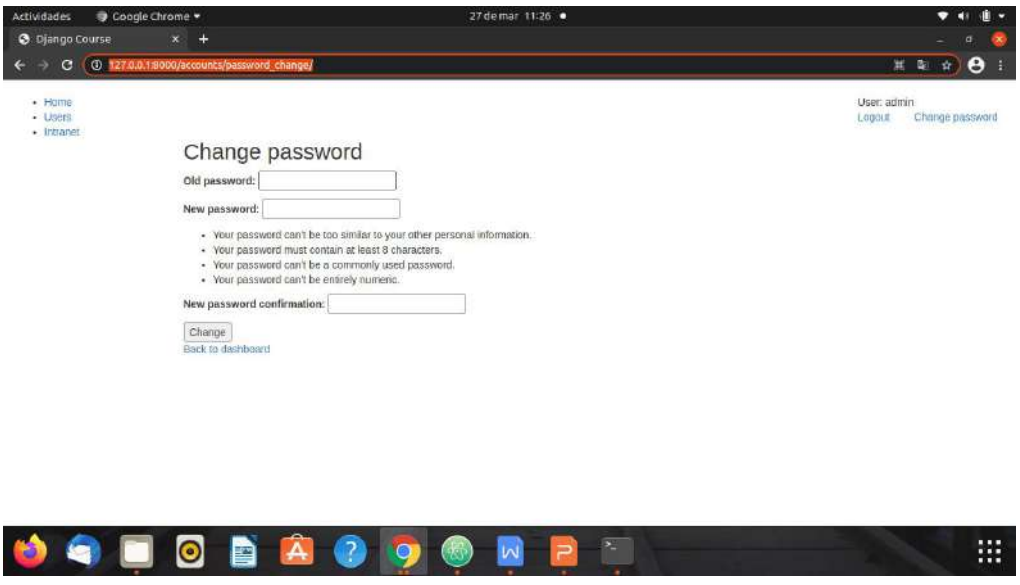
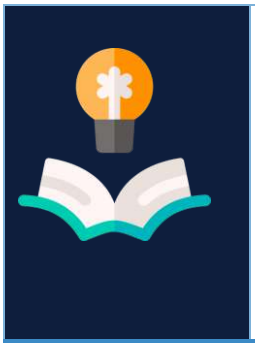


Figura 3.5: Imagen en la que se muestra la página donde se cambia la contraseña.



Recuerda

Django nos habilita un administrador para facilitarnos la gestión de usuarios y seguridad de nuestra aplicación.

4. PUNTOS CLAVE

- Django permite la gestión de usuario gracias a la librería: from django.contrib import admin
- Es necesario crear la plantilla, declarar la plantilla en la función python, declarar la función en la url y finalmente llamar a la url desde un botón para poder visualizarlo en el navegador